OULUN SEUDUN
AMMATTIKORKEAKOULU

Pauli Paaso

ROADMAP FOR SUCCESS

Agile testing methods

Pauli Paaso

Master's Thesis

Spring 2011
Degree Programme in Information technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Pauli Paaso
Title of thesis: Roadmap for Success
Supervisor: Tuomo Tikkanen
Term and year when the thesis was submitted: Spring 2011 Number of pages: 35

Software testing requires a tight focus. It's easy to try to test too much. Before you start to develop test cases in a system and, obviously, after you have taken the time to learn the product reasonably well. You need to figure out what you might test, then what you should test, and finally what you can test. Determining the answers to these questions and to find new perspectives for the planning of testing of software products in agile projects was objective for writing this Master Thesis.

This Master thesis provides directions to reach a desired destination, software is tested and accepted. Test strategy is highest level activity with software testing. Test strategy determines all of testing activities. Test plan Test plan describes how the testing will be done and what kind of environment, tools and methods will be used.

Software projects usually proceed with one of the selected process model. The most common process models split project to different phases. Project models can be used in several variations, but general in all models have separate planning, specification, design and implementation phases. All of these phases contain also testing activities. All project members have own role in project.

Software testing is and will continue to be a fundamental activity of software engineering. To engineering the test process, we need to collect evidences for such information to be able to find the most effective pattern for testing a system. This can routinely do, when for instance functional testing based on the requirements is combined with measures of code coverage adequacy. Practices need to be backed up by a systematic effort to extract and organize recurring and proved effective solutions to testing problems into test patterns, similarly to what is now a well-established scheme for design patterns approaches.

Keywords:

Software testing, Scrum, process model, testing levels, test plan, testing techniques

Table of contents

1 INTRODUCTION

I have worked in Symbio Finland Oy as a Software Test Engineer and a natural choice would be a theme in my case is the planning a suitability of software testing methods to a project. The strengths and weaknesses of different popular testing types are identified through a systematic literature review. The name of the work is ”Roadmap for Success” and the objective are to find new perspectives for the planning of testing of software products in agile projects.

Information about agile and project models is searched from literature. The agile software development methods are driving software engineering closer to the customer. Basic idea in the Iterative processes is to create software in small steps. If software product is done in this way, customer can try out the developed software and can make requirement changes and prioritization that are valuable for customer. This list is known as the Product Backlog. This brings new challenges to software testing process. Software product should be tested after every iteration round. To ensure software quality, the customer or product backlog owner must be used at every stage of the project. This helps develop the software that the customer really wants, and it minimizes the cost of waste work.

Test automation framework can be used for basic functionality testing. Every user story and use case scenario must be tested and verified with target device in live environment. Test cases have to be designed so that they cover user scenarios and all requirements. Testing is the processes of examining an application to ensure it fulfils the requirements for which it was designed and that it meets quality expectations. Testing ensures that application meets customer expectations. The purpose of this report is to highlight issues that need to be handled and resolved in the design of testing.

2. SOFTWARE PROJECT STEPS

Software projects usually proceed with a selected process model. The most common process models split a project into different phases. Project models can be used in several variations, but general in all models have separable planning, specification, design and implementation phases. All of these phases contain also testing activities. All project members have a role in the project.

2.1 Feasibility study

Software project life cycle usually starts with a feasibility study. Feasibility study report examines what customer and user requirements are to be developed commercial product. This phase determines the overall system level requirements of the main objectives, namely a list of product characteristics and features. The definition is also called customer requirement specification because it observes the customer and user needs. It does not comment actual implementation. The goal is to get the answer to the question of why such a product should be developed or, conversely, why it should not be implemented.

A feasibility study focuses on the study of the challenges, technical problems and solution models of information service realisation, analyses the potential solutions to the problems against the requirements. Feasibility study evaluates ability to meet the goals and describes software features. Requirements analysis rationalises the recommended solution [1].

ISO 12207 Standard for software lifecycle basis on five different roles. Acquirer role is in charge of the acquisition process, from the definition of requirements and the initiation to the control of the contractor up to the acceptance.

Supplier role is supply process. Supply process begins with the placement of the order and the signing of the contract and ends with the delivery. Developer role is actual

development process. Developer role can even include the system requirement analysis, and it ends with the installation and introduction of the system.

Operator role is operation process and comprises review, realization, and support of the operation. Maintainer role is maintenance process. Maintenance begins with modification and ends with migration and finally software retirement. Figure 1.1 shows steps from Feasibility study to requirement analysis. Table 1.1 shows an example of requirement table

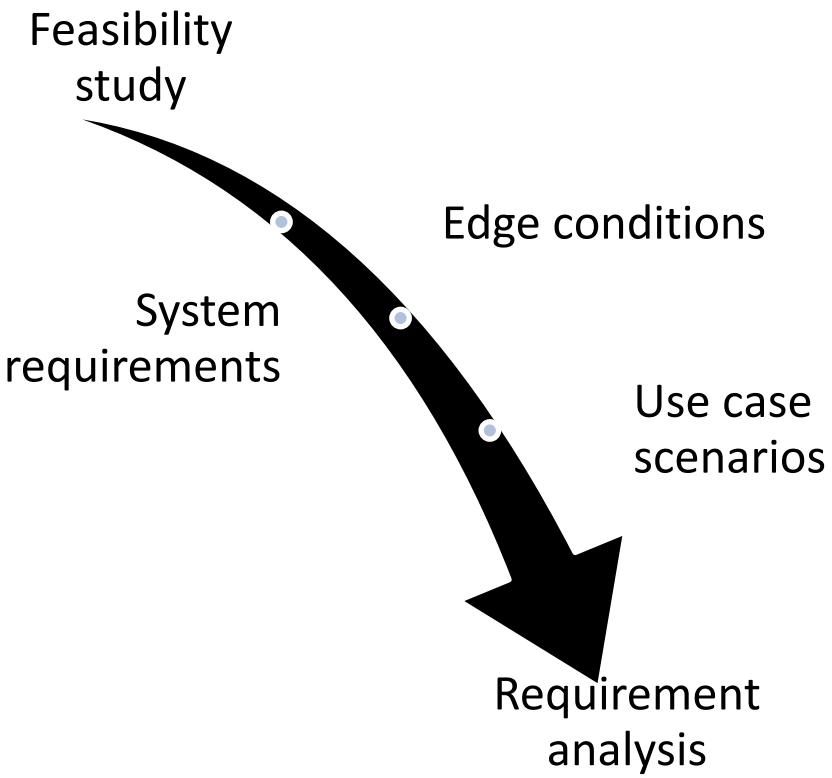

Figure 1.1 Feasibility study step to requirement analysis

Requirement ID	Old Requirement ID	Feature	Name	Description	Testable	Priority for testing
1001	25581	Persistent user data capability	User data management	Saves user data to safe area.	Yes	Priority high
1002	25582	Data synchronization with PC	Sync	USB 2.0 connection synchronization with PC	Yes	Priority normal
1003	25583	Software version update	Version updating	Software update from release repository	Yes	Priority normal

Table 1.1 Example requirement table from requirement analysis phase

2.2 Definition

The ISO 12207 standard defines all tasks required for developing and maintaining software products. System requirements are defined first. Software requirements analysis is the next phase after Feasibility study. The objective is to describe the product itself.

Software requirements analysis has been modified from customer requirements to accurate product specifications. Use case scenarios and implementation details are clarified. It is easy to implement something when you know you exactly what to do. Figure 2.1 shows a process of life cycle for software.

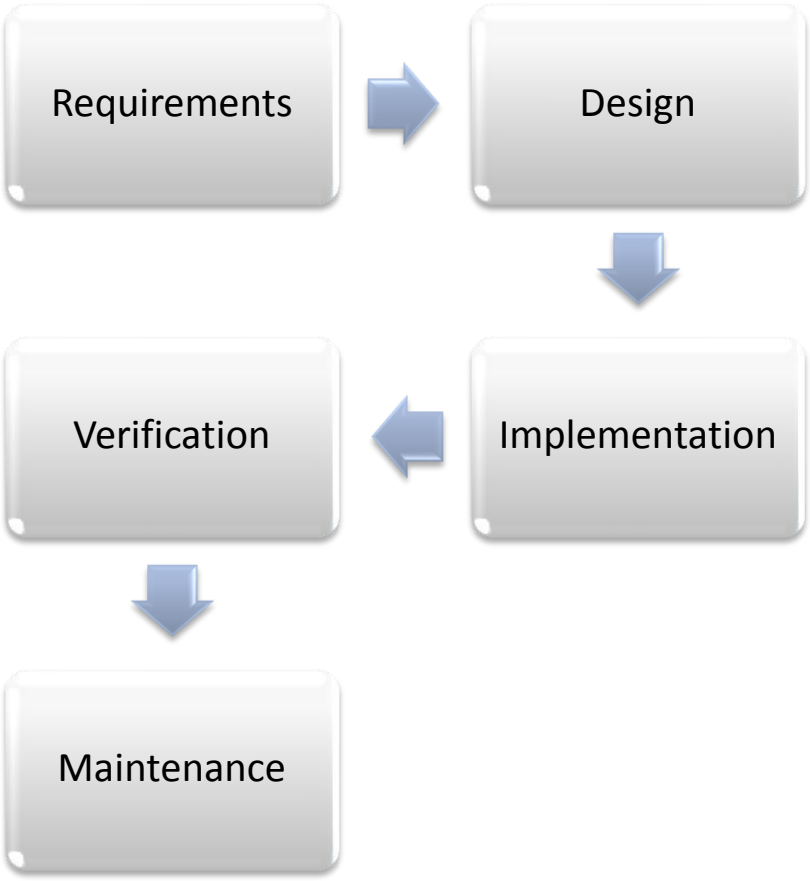

Figure 2.1 a process of life cycle for software

2.3 Design

Design and architecture are the activities involved in specifying how the software will actually work. This phase is called Engineering. How to implement specified application? This is a multi-level stage, which can be divided into four parts: data structure, software architecture, user interface, and operational details like interfaces to other modules.

The design phase can often consume most of the project time. Poor design can lead to failure of a project. The test planning and scheduling are also clarified in design phase.

Design patterns can speed up the development process by providing tested, proven development paradigms. Design patterns make easier to reuse designs and architectures. Figure 2.2 shows dependency structure design from software component point of view.

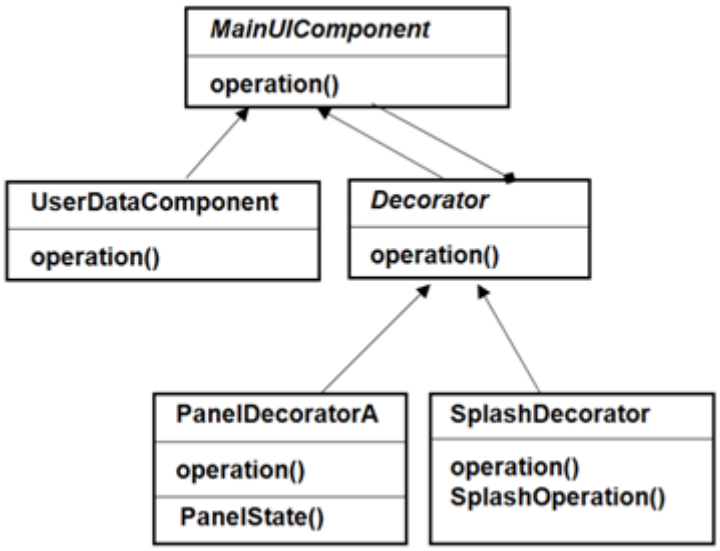

Figure 2.2 Dependency structure design from software components

2.4 Implementing and testing

After the design of software has been done, it is time for implementing. Implementation starts with interface definitions according to the design documents. Design and implementation must be monitored and if necessary they may be edited. One major source of error may be that the product specification and design are not sufficiently aware of the development cycle. When the program is ready and functional, some of features can be tested. The test environment and schedules of testing of are clarified. Functionality, UI correctness, interaction with other applications, correlation to generic requirements and memory leak checks must be done. When the requirements and boundary conditions are met the software quality, have to go the next stage of software development process.

3. SOFTWARE PROJECT PROCESS MODELS

Software projects usually proceed with a selected process model. Next, we move to examine the most commonly used project models.

3.1 V -model

The name to model comes from Verification and Validation process. The process emphasizes requirements-based design and testing activities. All design elements and acceptance tests must be traceable to one or more of system requirements. The model is based so that time and maturity move from left to right and one cannot move back in time. V-model is easy to use if requirements are known and freeze.

This model supports project tailoring. V-Model 97 is a German government software development standard for IT projects. It is often mandatory in software development contracts with the German Federal Office of procurement. The VM97 is compatible to quality management standards GAM-T17, US-DOD-498, ISO 12207, ISO 900x and STANAG 4159 [6]. Figure 3.1 shows V-process model description [6].

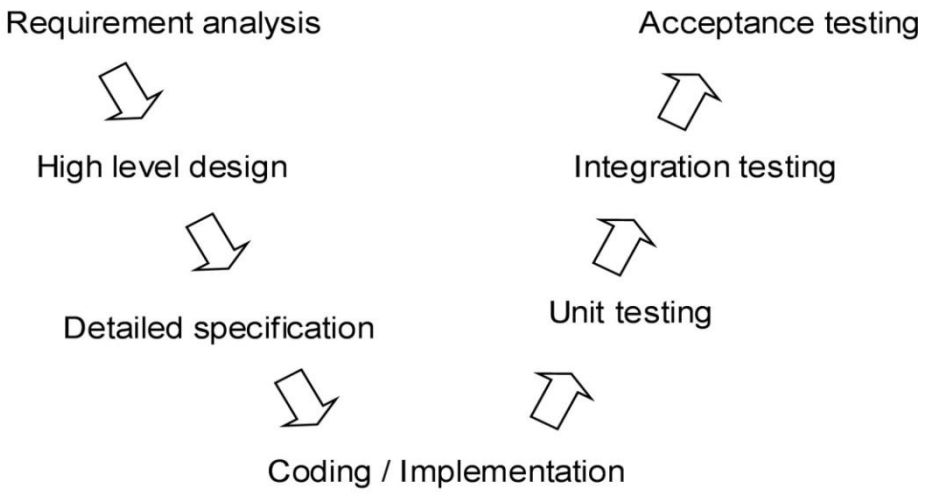

Figure 3.1 V-process model descriptions

3.2 Waterfall

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards like a waterfall. The model shows how each major development phase progresses. Winston W. Royce was the first who described Waterfall model [2]. The model is useful in helping developers lay out what they need to do.

A problem with waterfall model is that none of the phases of the waterfall project is not really going to complete in first iteration. This model is not good if requirements are changing. This model can be used when customers know exactly the requirements and are willing to wait that software systems will be ready for release (military and government). The cycle from requirements to product can take years. This model is slow for changes. Figure 3.2 shows Waterfall process model description. Requirements must be very well defined. The design phase must allow for sufficient time.

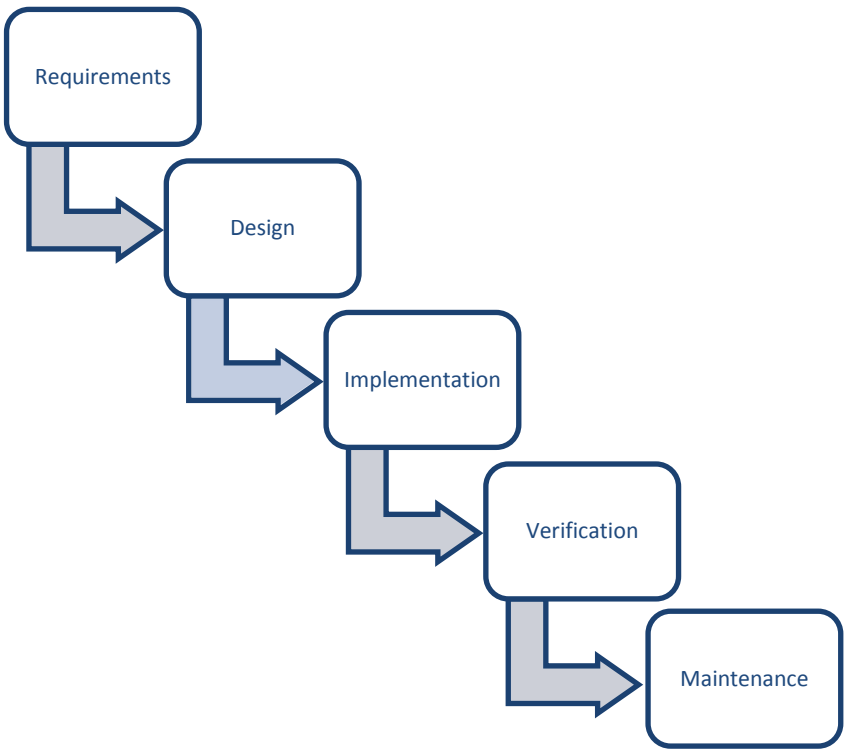

Figure 3.2 Waterfall process model

3.3 Scrum

Scrum is an iterative methodology for project management. Scrum process contains sets of practices and predefined roles [3]. The main roles in Scrum are Scrum Master, Product Owner and the Team. The idea in Scrum process is to create software in small steps. When a software product is done in this way, the customer can try out the developed software and can make requirement changes and prioritization that are valuable for the customer.

The product owner creates from the software requirements analysis report a prioritized wish list called a product backlog. This backlog is dynamic and items may be deleted and added at any time during project. The team has a certain amount of time for a sprint to complete its own sprint backlog. Items are moved from Products owners' backlog to sprint backlog in sprint planning session. The sprint starts with a sprint planning session.

After the planning session items in the sprint backlog items are estimated by team members in an estimation session. Every item get own score or effort estimation total task hours to complete single item. All items are estimated and added together and overall effort estimation is calculated for sprint. A burndown chart shows work remaining tasks over time. Figure 3.3 shows example of a sprint burndown chart. In example chart all items for a sprint are estimated for 450 hours. If there are no impediments for any item, sprint ends in day 10.

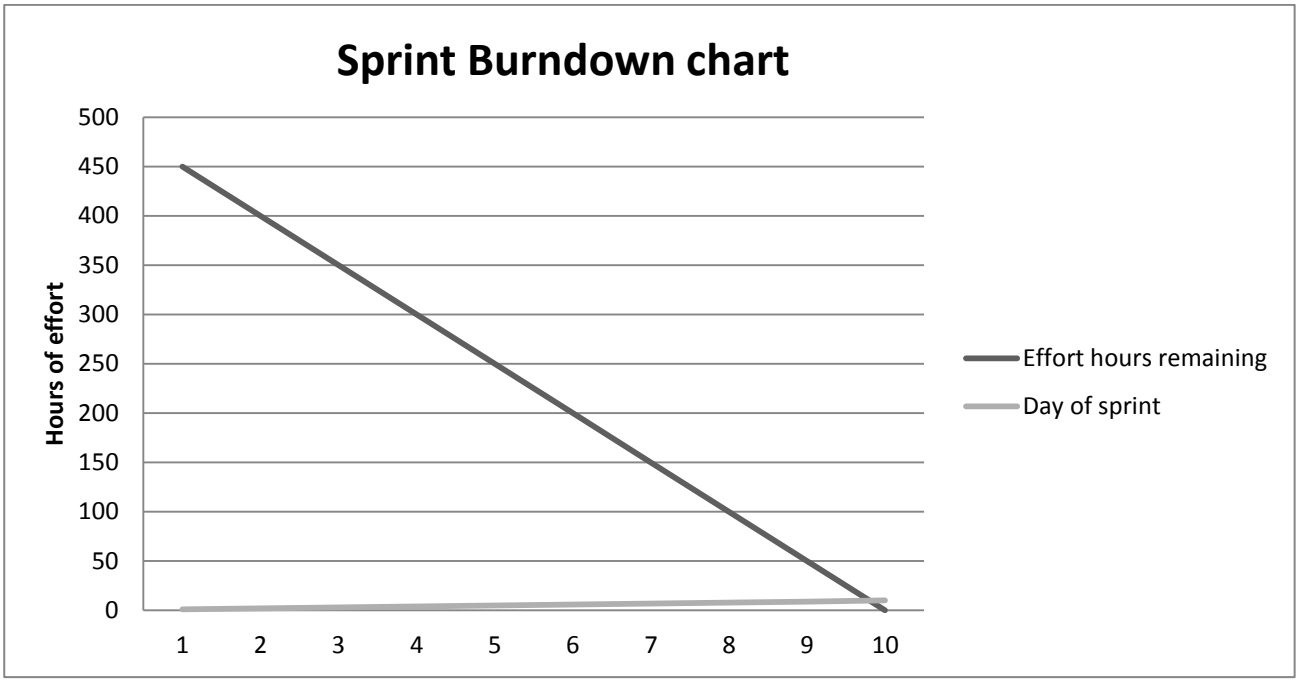

Figure 3.3 Scrum process overview

Sprint work usually takes two to four weeks. Figure 3.4 shows overview of Scrum process. The development progress is monitored with daily Scrum meetings. The Scrum master's role is to hold daily Scrum meetings and keep the team focused on its goal. Scrum Master takes care of impediments with a team and keeps up progress and update the sprint backlog and burndown chart accordingly.

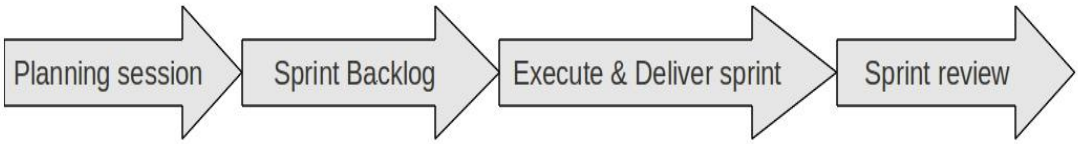

Figure 3.4 Scrum process overview

The sprint ends with a sprint review and retrospective. The product owner accepts done Sprint Backlog items. At the end of the sprint results are shown in the sprint review hosted by the Scrum master. The purpose of the sprint review is to demonstrate developed and tested functionality to the product owner and the stakeholders.

Figure 3.5 shows this phase of Scrum process model. The retrospective is for the Scrum team. In retrospective the team discusses three things: what went well, what didn't go well, and what improvements could be made in the next sprint. The product Owner doesn't attend to this meeting and it's an opportunity to speak freely about successes and failures. This is an especially important opportunity for the team to focus on its overall performance and identify strategies to improve processes. The team can look back on what they have accomplished together and be open about how they can work even better.

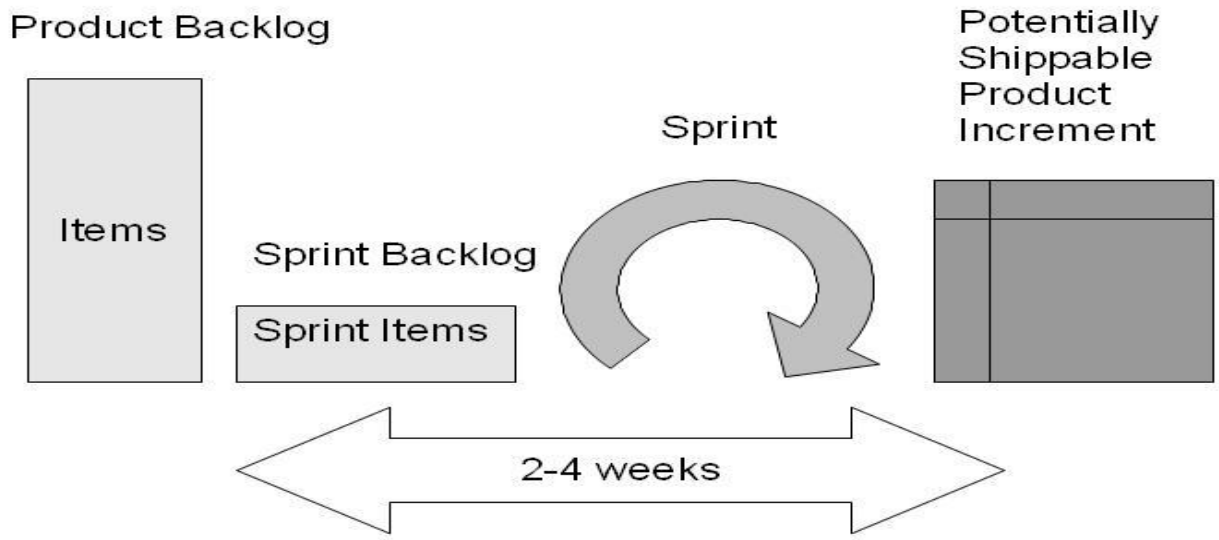

Figure 3.5 Scrum process model

4. TESTING WITH PROCESS MODEL

Testing requires more than just executing a set of tests. The testing process is divided into stages and levels, as with any software development phase. In the design phase it should be considered what methods are recommended and suitable for this particular project, as well as which tools would support testing process. It's easy to try to do too much, but everything can’t be tested. It takes time to get to know the product reasonably well before starting to develop a test system. You need to figure out what can be tested and what are testability and scheduling aspects. After finding the answers to these questions you can focus on test efforts and test what should be tested [4].

4.1 Basic Idea of software testing

An error or a mistake in software development process produces incorrect result. This can cause a flaw in code and bug fails to perform required function. If a defect is in a software product, then system may produce an incorrect result. The objective in testing is to discover these bugs or strange behaviours.

Starting the testing and verification process as early as possible saves a lot of effort and money. If an error is found at an early stage, it is easier and cheaper to repair it. For this reason, it is good to establish test documentation after each release. As in other phases of the project, a software product must be tested. So you have taken into account that agile process doesn’t need documentation, scope, content and scheduling. The test documentation is intended for software testing team, development team and project management for the purpose of sharing information about maturity level for developed software and need for refactoring software components.

4.2 V-model in practice

In the V-model software is designed on the left hand part of the model and built and tested on the right hand side. Thus testing and defects in the model are found later than in the iterative process model. Validating requirements may seem tricky given that if there are no specifications in first phase and acceptance testing is not possible to execute. Smoke testing is a testing type to start up a project and early testing with V-model. The idea is just to start the application or system and move around in it to see what it is and what it can do. Testing in V-model is like a gatekeeper. If software works like it is required, specified and designed, it can be released. The V-model is a much slower process than the Scrum. The V-model is well suited to projects where the requirements do not change.

4.3 Scrum benefits and known problems

Agile methods use short iterations, which means that in short periods of time, working software is released so that the stakeholders can check if the implemented software is exactly what they need. Stakeholders can make requirement chances and can make prioritization of features that are valuable for the customer. Software product should be tested after every iteration round. Known problems in Scrum and iterative models are incompatibility between modules, broken dependencies, out of date modules, low test coverage and lack of compliance to coding standards because of short period of implementation. Integration with version control is also more complex due to a software component interface changes.

From testing point of view the best testing method to fit agile processes is exploratory testing. The main advantage of exploratory testing is that less preparation is needed, important bugs are found quickly at short execution time period. Exploratory testing is good for Scrum because it needs minimum amount of planning. It may be problematic to find time box for a test before new sprint starts. There may be challenges like if there is no

good enough code to be tested after the first sprint. If lots of bugs are found during sprint, when is time to correct them? Another good testing method for Scrum is regression testing the purpose which is to ensure that existing functionality is not broke up with new implementation.

4.4 Waterfall benefits and known problems

In a Waterfall model, after each phase is finished, it proceeds to the next one. Reviews may occur before moving to the next phase which allows for the possibility of changes. Reviews may also be employed to ensure that the phase is completed. The phase completion criteria are often referred to as a "gatekeeper" that the project must pass through to move to the next phase.

Waterfall discourages revisiting and revising any prior phase once it's complete. A Waterfall model has been a source of criticism by supporters of other more flexible project models.

5. TESTING PROCESS

Software testing process starts when the software development project starts and it is a continuous process until the software is retired.

5.1 Software Testing types

There are many different types of software testing techniques available. There are two main categories of testing, static and dynamic. Dynamic techniques are divided in specification based, structure based and experience based. Static techniques do not execute the code. Static testing can be done with code reviews manually or with tools. Specification based testing is called black box testing.

Structure based testing is called white box testing. Non-functional aspects of software are measured with performance metrics, usability, maintainability and stress tests. Structure based testing techniques uses the structure of the software to derive test cases. They require knowledge of how the software is implemented.

5.2 Test Planning for a project

Test planning determines testing scope, risks and objectives of testing. Testing strategy defines test levels and entry and exit criteria. The objective of testing depends on the level of testing. A phased test approach detects bugs more efficiently. The purpose is to specify the testing scope, testing approaches, allocation of resources and scheduling of the test activities. Test plan also provides the details of the product being tested. It defines features to be tested along with the resources responsible for the completion of each testing task. Testing may be performed with varying degrees of formality. Formal testing process uses formal test cases with extensive documentation. Informal testing process uses only test

case name and purpose of test case documented.

5.3 Test levels

The lowest level of testing is unit testing. It tests the basic unit of software, which is the smallest testable piece of software. In object-oriented programming a unit is usually a method. Ideally a test case is independent from the other units or modules. Idea for Unit testing is isolation. Stub objects fakes tested unit and test harnesses can be used to assist testing a module in isolation. The purpose of unit testing is to isolate each part of the program and prove that the individual parts are correctly implemented and works like it is designed.

A unit test can be seen as a design element specifying classes and methods. Test case observes unit behavior. Testing will be done by the developer and will be approved by the development team leader. Proof of unit testing (test case list, sample output, data printouts, incident information) must be provided by the developer to the team leader before unit testing will be accepted. All unit test information will also be provided to the test documentation.

Component Testing is performed when two or more tested units are combined together. Component testing is often done on both the interfaces between the components and for the larger structure.

Integration testing is performed to verify functional, performance, and reliability requirements placed on major design software component items. Designed component items are exercised through their interfaces. Success and error cases being simulated via appropriate parameter or data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components interact correctly.

System testing is performed for end-to-end quality of the entire system. System test is based on the functional requirement specification of the system.

Acceptance testing perform a Non-functional quality attributes, such as reliability, security, and maintainability. The purpose of acceptance testing is rather to give confidence that the system works as expected.

Pilot testing is used for testing a product with potential user group before it is launched to market. The main purpose of pilot testing is to catch potential problems before they become costly mistakes. Usability and user experience testing can be part of pilot testing. Figure 5.1 shows all testing levels.

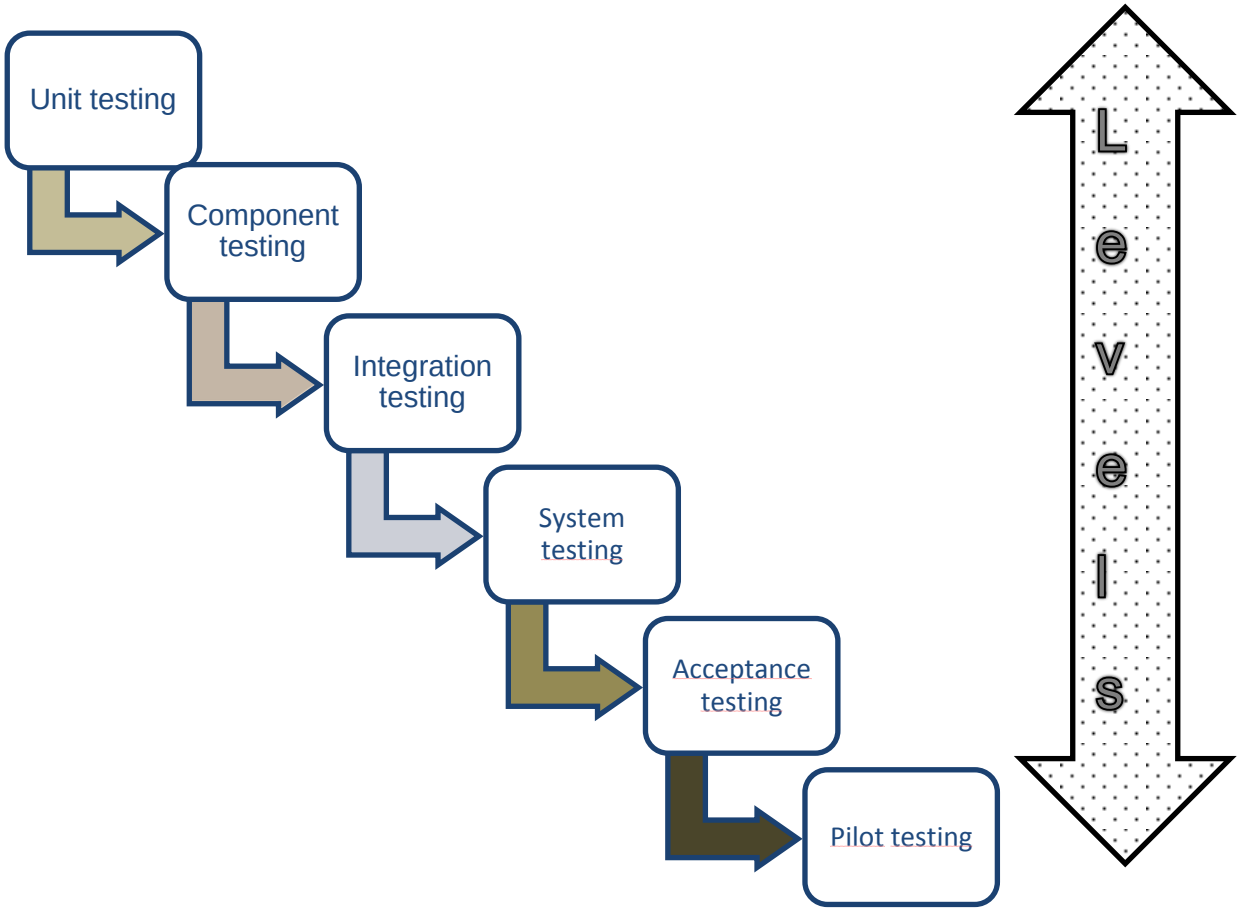

Figure 5.1 Testing levels

5.4 Test Scheduling

Before we can execute any tests, we need to know what we are doing. Test scheduling starts from analysing of test basis in the beginning of project. If we have requirements and specifications or business process descriptions we can plan test scheduling. We must know also how the testing will be done [3]. Test conditions, levels and procedures must document to Test Design Specification. First possible software testing level is unit level. In component level unit testing framework must be planned to use.

All testing activities must be controlled and scheduled with relevant milestones. Milestones should be identified with their relationship to the development process. Time must be allocated within the project plan for the following testing activities. The specific dates and times for each activity are defined in the project plan time line. The persons required for each process are detailed in the project time line and plan as well. Coordination of the personnel required for each task, test team, development team, management and customer will be handled by the project manager in conjunction with the development and test team leaders.

5.5 Test Termination

How much testing is enough? This is question where answer is a decision that based in a number of factors. Most important factor is risk that there are critical potential defects in software. Testing reduces the probability that undiscovered defects are remaining in product software. The project schedule must have the time box for testing. Sometimes bug related exit criteria such as no open high severity bugs works. Basically, whether or not to ship a product is a business decision, and some amount of flexibility is required.

5.6 Bug correction process in Scrum

After bug is found it must be documented. Every new release of a software or hardware component should have a release number or identifier attached. This identifier is essential for determining which version of the system contains a bug.

Document for describing bug is called Incident report. Incident report is giving feedback about problem that has found to development team. Information we should record to incident report is test id, details, software version, hardware type, hardware version, actual result, expected result, severity, priority, and reproducing steps.

After reporting Bug is at open state. All open bugs must add to sprint backlog. After adding open bug to backlog item developer team or some of team members take it to responsibility. Development team fixes bug and makes a new release of software. Bug state changed to corrected state.

Testing team test that bug doesn’t occur and makes verification with new release. If fix is good and software works, state must be changed to Verified. After that verification bug can closed. Scrum master or test manager can close bug. If fix doesn't work, bug can be reopened state and process starts again. Figure 5.2 shows bug correction phases.

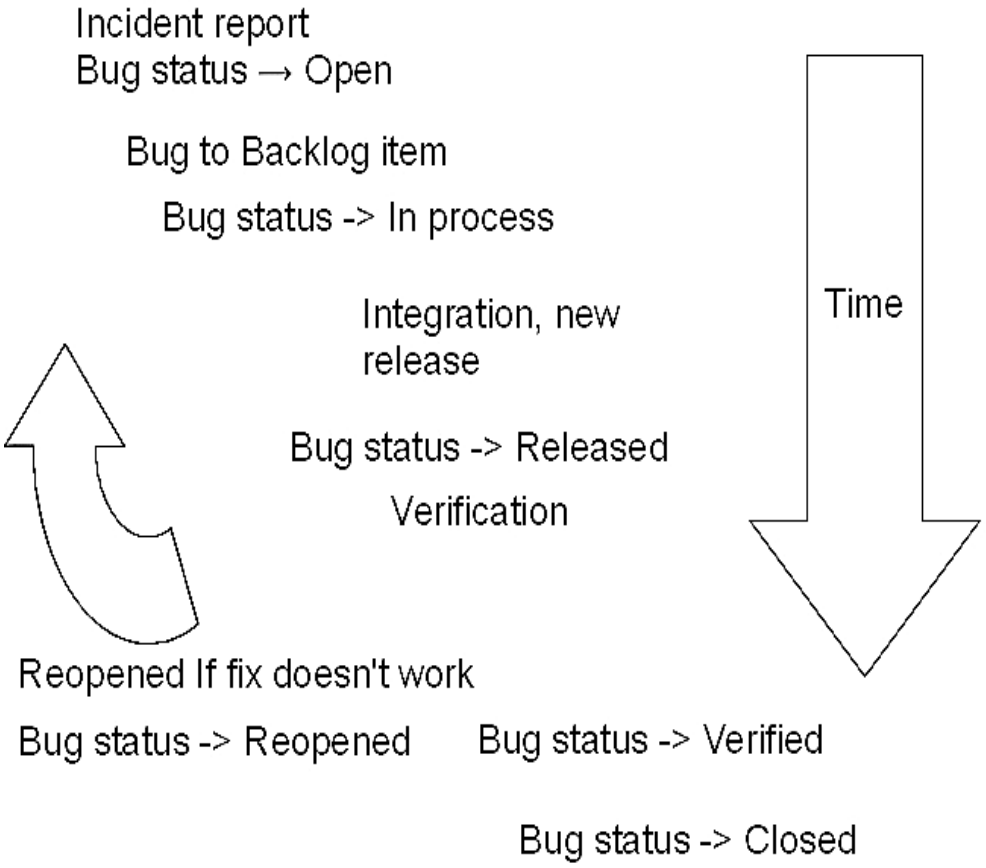

Figure 5.2 Bug correction phases

5.7 Roles for testing

Structural tests like unit testing is typically conducted by the development team members. The objective of structural unit testing is to find bugs in low level operations. Integration testing objective is test module interfaces and how they interact together. This is done by development team or integration team. System Testing test functional and structural stability of the system by the test team.

Systems Integration Testing is to provide confidence that software is able to interoperate successfully with other specified software systems. This is done by test team. User Acceptance Testing is to provide confidence that the system works correctly and is usable before it is formally delivered to the end user. Testing can be done by end users with customer scenarios. Hardware development often involves high level pilot testing, either following or in parallel with acceptance tests.

Pilot testing checks the ability of the assembly line to produce production for mass product the finished system. This phase can include customization of software development, where it demonstrates to potential business customers that the system will perform all the necessary operations in a live environment.

6. TEST DOCUMENTATION AND REPORTING

Software testing activities must be documented. IEEE 829 standard specifies test documentation. Test plan specify features to be tested and features not to be tested and overall test strategy. Test Design Specification document is to provide an intermediate level view of the testing process for one or more product, subsystem, or component features or attributes. Test case specification specifies test preconditions, step by step details what feature is tested and details how it is done and post conditions and expected result.

Test Incident Report: detailing, for any test that failed, the actual versus expected result, and other information intended to throw light on why a test has failed. The reason is that a discrepancy between expected and actual results can occur for a number of reasons other than a fault in the system. These include the expected results being wrong, the test being run wrongly, or inconsistency in the requirements meaning that more than one interpretation could be made. The report consists of all details of the incident such as actual and expected results, when it failed, and any supporting evidence that will help in its resolution. The report will also include, if possible, an assessment of the impact of an incident upon testing.

6.1 Test planning document

Test planning is related to the management of testing process. Test plan describes how the testing will be done. Test planning should be overall test strategy and test policy. It defines what will be tested and who will be done it. What is test coverage and what kind quality level is required. What kind of tools will be used for testing? There is two approaches to plan testing activities: preventative and reactive. Preventative approach involves much planning and preparation work. Testing can start after software comes available. Reactive approaches wait until software comes available. Test plan produced specific scope and a level of testing. Test approach is overall test strategy. Are any special tools to be used for

testing and what are they? Do tool require special training? What metrics will be collected? Which testing level is each metric to be collected at? How is Configuration Management to be handled with this project? How many different hardware configurations will be tested? How many hardware and software and Combinations are planned to test. What levels of regression testing will be done and how much at each test level? Will regression testing be based on severity of defects detected? How will elements in the requirements and design that do not make sense or are untestable be processed? There are a lot of questions to be processed before testing can be performed. Table 6.1 shows example of testing tool selections.

Tool's name	Purpose	Coverage
UI test framework	Test UI functionality	95,00%
PyUnit	Test Automation framework	95,00%
GCov	Code coverage tool	95,00%
CMock	Module/object mocking framework	100 % API*
Bugzilla	Defect Tracking System	

Table 6.1 Example tool selection table

Manual tests may find many defects in a software application but it is a laborious and time consuming process. In addition, it may not be effective in finding certain classes of defects. UI test framework is a test automation tool for purposes to writing a computer program to do testing that would otherwise need to be done manually. PyUnit is an easy way to create unit testing programs and Unit tests with Python. The Gcov -coverage testing tool analyses the number of times each line of a program is executed during a run. This makes possible to find areas of the code which are not used, or which are not exercised in testing. When combined with profiling information to the information from coverage testing allows efforts to speed up a program to be concentrated on specific lines of the source code.

CMock is a module or object mocking framework for C projects. It generates dummy modules that conform to the interface specified in a header file. This allows APIs to be proven out and exercised before committing to an underlying implementation.

Bugzilla is a web based defect tracking system or bug tracking System. Defect Tracking Systems allow individuals and groups of developers to keep track of outstanding bugs in their product effectively.

6.2 The test cases

This is a part where test objectives are translated to test conditions and designed to cover most important test conditions. Test case implementation can start. In component level unit test should test every interfaces, classes, methods and variables. Integration level test should test module interfaces with other modules, system performance, and reliability. Test cases are constructed to test that all components interact correctly procedure calls or process activations. In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement. One case is positive happy scenario. Another one is negative scenario. Table 6.2 shows example of functional test case of web based system UI-level testing.

Test case name	Account - Sign In
Test case ID	3056
Description:	User has account to web service and try to log in with wrong username or credentials. The web server does not accept the logon, but gives info note: 'Username or password do not match. Please check your input.'
SW Release:	0.9.288.1
Logical Component:	Login module
Test Case Priority:	High
Comment	Important to work

Table 6.2 Example test case

Every test case is identified with unique “short” name for the case. IEEE 829 standard have template for test case specification. It's a good idea to name the test case with header text and numeric identifiers. The header is a short description, only a few words long, that conveys the essence of the test. For the numeric identifier a test case that is used in test suite 2 and that is the second test case is assigned identifier 2.2.

The next entry in the header lists the name of the test suite in which the test case will be used. Because a given test case might be used in multiple suites, this entry could get a bit unwieldy. But in practice most test cases are used in only one test suite, so including the name of the suite provides some useful information with only infrequent confusion. Prioritizing test is most useful when you need to determine how many times a given test should be run. Priority based on the opinions of the sales, marketing, technical support, and development team and from coverage analysis or all of these. Test coverage analysis, for example, allows you to assign the highest priority to those test cases that cover the most important quality risks, requirements, and functions.

The next entries in the header address resource requirements. For the first two of these entries, you should list, row by row, the hardware and software needed to run the test. The entries for duration and effort specify how long it will take to run the test, in clock time and in person hours, respectively.

6.3 The test summary document

The Test Summary brings together all information about the testing. It describes how well the testing has been progressed, the number of incidents raised and outstanding, and crucially an assessment about the quality of the system. This document is important in

deciding whether the quality of the system is good enough to allow it to proceed to release.

In a summary is overall assessment of all test cases. The status entry should be marked pass, warn, or Fail, depending on the success of test case. Document shows overall pass rate. Pass rate is calculated successfully completed test cases divided number of all cases. Value is percentage form.

I think that this is not enough to guarantee the real usefulness and adequacy to purpose of the tested software: as importantly, behaving as expected software must fulfil also functional properties, depending on the specific application domain. While conventional functionality testing does not provide for any notion of time, many features of the exhibited behaviour of a piece of software can depend on when the results are produced, or on how long they take to be produced. Similarly, while functionality testing does not tackle resource usage and workloads, in specific domains, such as telecommunications, performance issue account for a major fault category [5].

7 CONCLUSIONS

Software testing is difficult and richly articulated research discipline, and hope that this Master Thesis has provided a useful overview of current and future challenges. What is assured is that software testing engineers do not risk remaining without their job.

Software testing is and will continue to be a fundamental activity of software engineering. To engineering the test process, we need to collect evidences for such information to be able to find the most effective pattern for testing a system. This can routinely do, when for instance functional testing based on the requirements is combined with measures of code coverage adequacy. Another recurring recommendation is to combine operational testing with specific verification of special case inputs. Practices need to be backed up by a systematic effort to extract and organize recurring and proved effective solutions to testing problems into test patterns, similarly to what is now an established scheme for design patterns approaches.

A roadmap provides directions to reach a desired destination, software is tested and accepted. Test strategy is highest level activity with software testing. Test strategy determines all of testing activities. Test plan Test plan describes how the testing will be done and what kind of environment, tools and methods will be used. When an exit criterion is achieved, test summary report can be composed.

Successful tests are those that find problems, problems that often occur due to errors in the timeless and common failure modes. Some problem areas are the classic errors in the implementation of conceptual models are the classic errors associated with them, some technologies have problems or limitations, some failures are critical or important, and they must be validated, and some mistakes in handling consistently been neglected.

Test design, expert testers often rely in part on experience things that have gone wrong in the past. This experience does not include any functional specification of the product, and

therefore is not called explicitly verified. Experience expert tester can be captured and turned into a valuable test design heuristics.

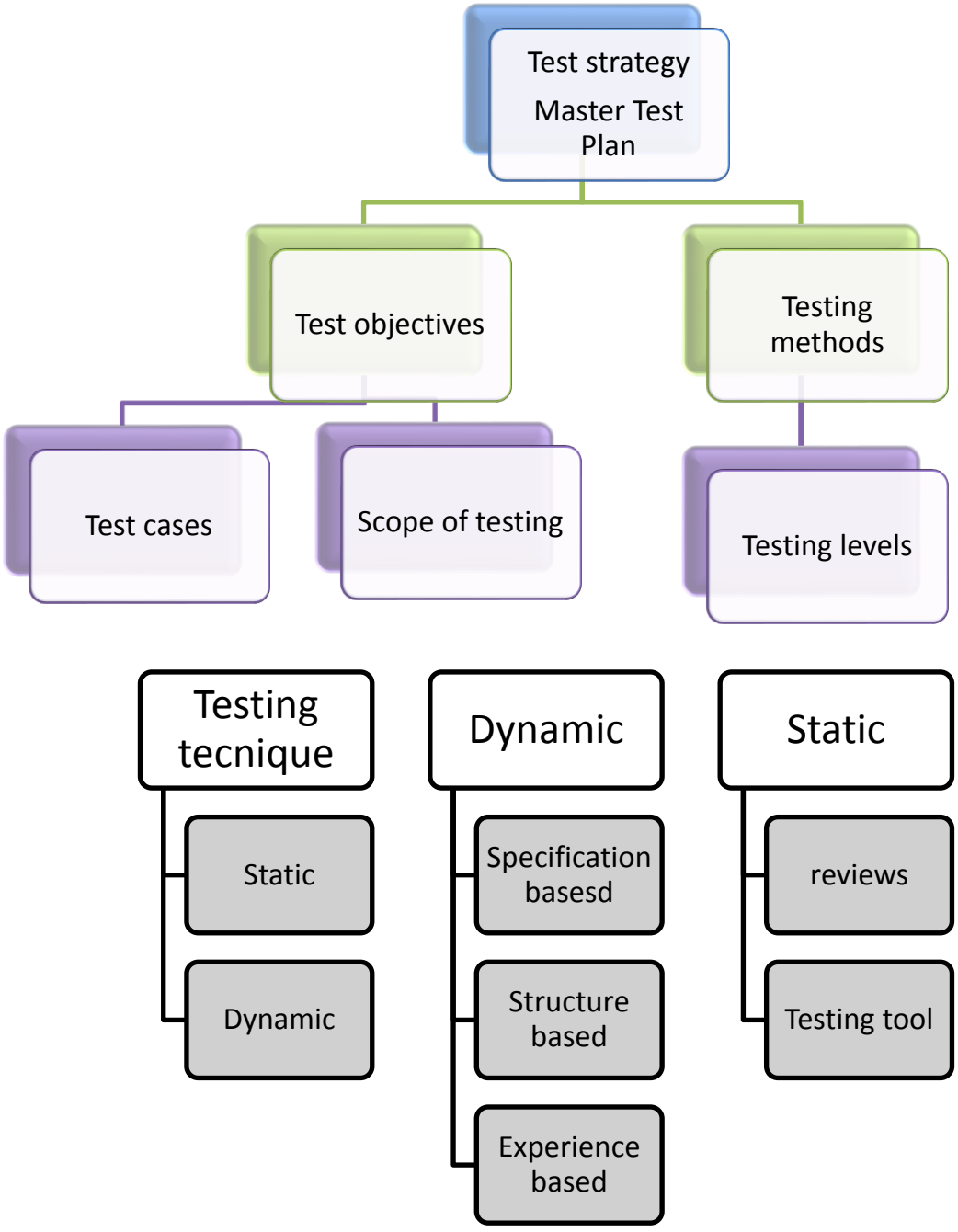

Figure 7.1 a roadmap for successfully testing

A roadmap for Success Master Thesis provides also template for Test Plan. Test plan should contain following chapters.

Test Plan Template content:

1 Overview

Overview of software, what will be tested, device under test and main features list. Dependencies to other software components are defined.

2 Used Test levels and phases

Testing levels are defined and testing must follow development cycles.

3 Purposes and Scope

This chapter gives the overview on the features to be tested and the features not to be tested. This chapter defines Scope for testing.

4 Definitions

How software should behave in different situations. Use case scenarios are defined. Use case scenario can be test case. User Interface specification can be used for test case implementation.

5 Setting test configurations and environments

Target end device and testing environment settings are defined this chapter.

6 Quality Risks

Product risks for software that all bugs aren't found during testing are listed under this chapter.

7 Proposed Schedule of Milestones

Test schedule must follow project scheduling. Test schedule is defined and fixed.

8 Entry Criteria

Software development cycles and releasing schedule is defined.

9 Stopping Criteria

When testing can't be performed this stopping criteria defines conditions or events that would lead to suspend test execution. If test environment become unstable or the system might have so many bugs open or such severe bugs that it makes no sense to continue testing.

10 Exit Criteria

Exit criteria for testing are defined. All the planned test cases and the regression tests have been run and results are acceptable.

12 Test execution and test requirements

Test case descriptions and test cycles are defined under this section.

14 Resources

Responsible Test Engineer or test team allocation for project.

15 Tracking and Management of Tests and Bugs

What tools will be used and error management process defined here.

16 Bug Isolation and Classification

Bug severity levels and classification specifications defined in this section.

17 Release Management

What kind of software components are released and where is latest official releases.

18 Test Cycles

How long is test cycle and when test cycle is performed?

19 Referenced Documents

Reference document list

8 REFERENCES

[1] John W. Brockhouse Jr. and James J. Wadsworth, Vital Steps: A Cooperative Feasibility Study Guide

Available on the Internet

http://www.rurdev.usda.gov/rbs/pub/sr58.pdf

Date of data acquisition 3 march 2011

[2] Dr. Winston W. Royce, Managing the development of large software systems, IEEE Wescon, August 1970, The Institute of Electrical and Electronics Engineers

Available on the Internet.

http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

Date of data acquisition 15 April 2011

[3] Scrumalliance, Scrum Is an Innovative Approach to Getting Work Done

Available on the Internet

http://www.Scrumalliance.org/pages/what_is_Scrum

Date of data acquisition 23 February 2011

[4] IEEE Standard for Software and System Test Documentation

Available on the Internet

http://ieeexplore.ieee.org/servlet/opac?punumber=4578271

Date of data acquisition 23 February 2011

[5] ISO/IEC Standard 9126 Software engineering Product quality, Part 1: Quality model

Available on the Internet

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749

Date of data acquisition 25 February 2011

[6] Wolfgang Kranz, An Integrated System Development Process including Hardware and Logistics based on a Standard Software Process Model

Available on the Internet

http://ftp.rta.nato.int/public//PubFullText/RTO/MP/RTO-MP-102///MP-102-04.pdfDate of

Date of data acquisition 07 March 2011